# Towards the Automatic Restructuring of Software Requirements Specifications to Conform to Standards Using Large Language Models

Ryu Okamoto
*Graduate School of Information Science and Technology*
*The University of Osaka*
Osaka, Japan
r-okamot@ist.osaka-u.ac.jp

Shinji Kusumoto
*Graduate School of Information Science and Technology*
*The University of Osaka*
Osaka, Japan
kusumoto.shinji.ist@osaka-u.ac.jp

***Abstract*—Software requirements specifications (SRS) are essential to the success of software development. It is widely recognized that the quality of SRSs affects both the quality of their product and project results. IEEE 830 and ISO/IEC/IEEE 29148 are the international standards for the requirements process and SRSs. These standards provide recommended section structures and content of SRSs. However, a study has shown that many organizations have not adopted these standards due to a lack of knowledge or cost constraints. Indeed, there is a wide variety of SRS structures. This paper explores an approach for automatically converting SRSs with any structures into the standard structure. It is considered beneficial to standardize SRS structures. A key benefit is that it facilitates research on SRSs. It makes the assumption that SRSs in research follow the standard structure more realistic. As a matter of fact, some studies have assumed that sections of a SRS correspond to those of the standard structure. To achieve SRS structural standardization, we use a large language model (LLM) that demonstrates high natural language processing capability. We designed simple prompts and assessed the feasibility of an LLM. As a result, standardized SRSs contained about 80% of the required items in the standard structure, but they could not fully retain all information from the original SRSs. Observations of the results suggest that prompts should be designed to consider the size equivalence before and after standardization and relations between sections.**

***Index Terms*—requirements engineering, software requirements specification, SRS, restructuring, international standards, ISO/IEC/IEEE 29148, large language models, LLM**

## I. Introduction

A software requirements specification (SRS) is a document that delineates the specifications of its software. SRSs are essential for successful software development. Studies have reported that the quality of SRSs impacts both the quality of their products and project results [1][2]. Many studies on SRSs have been conducted with natural language processing (NLP) techniques [3] because SRSs are often written in natural language like English. Recently, the number of studies utilizing large language models (LLM) that achieve high performance in NLP tasks has been increasing [4].

ISO/IEC/IEEE 29148 [5] and IEEE 830 [6] are the international standards for the requirements process and SRSs. They provide the required quality attributes and recommend section structures for SRSs. Franch et al. surveyed the knowledge and use of these standards in industry [7]. They have reported that slightly less than half of requirements engineering practitioners are unaware of the standards. They also have found that even those who know the standards do not often use them due to a lack of knowledge, organizational culture, and cost. Indeed, even if we limit ourselves to the structure of SRSs, there is a wide variety in its form.

It is considered beneficial to standardize the structures of SRSs. Standardized SRSs can support communication among internal and external stakeholders. Standards provide a common language and unified way of understanding. In the survey by Franch et al., it was also mentioned that standards can facilitate communication with the involved parties. In addition, SRS structural standardization enables the application of SRS research findings. In fact, some studies have assumed that SRSs adhere to the standard structure or that sections of SRSs can be associated with sections in the standard structure [8][9]. Aoyama and Nakane have proposed a method for analyzing the quality of SRSs and an automated tool for it [8]. Although their method accepts structural diversity in SRSs, its application assumes that each section in target SRSs is mapped to a section in the standard structure. Chikh and Aldayel have defined a schema for managing traceability, targeting SRSs that follow the structure recommended by IEEE 830 [9].

Therefore, in this study, we explore the new task of automatically restructuring SRSs to conform to the standard structure. We use an LLM and assess its feasibility. Two key requirements for standardized SRSs are retaining information from their original SRSs and being organized appropriately according to the standard. We applied standardization to actual SRSs and assessed the results from these two perspectives. We evaluated the standardized from the first perspective based on automatic measurement of semantic textual similarity. For the second perspective, we manually evaluated them using a checklist of items that should be described in standardized SRSs. As a result, the simple prompts we designed did not fully retain information from the original in the standardized. On the other hand, the checklist coverage rates are about 80%, indicating that the LLM was able to restructure SRSs faithfully

to the standard. Furthermore, observation of the standardized suggested the need to design prompts that consider the size equivalence of the standardized and the original, as well as relations between sections.

The code used in IV and V, including datasets, prompts, and evaluation scripts, is available on Zenodo[1]

## II. Related works

ISO/IEC/IEEE 29148 [5] and IEEE 830 [6] are the international standards for SRSs. This study adopts ISO 29148 since IEEE 830 is superseded by it. Figure 1 shows a standard structure of sections in an SRS provided by ISO 29148. ISO 29148 also describes what should be written in each section except for section 3.4, *Interface Requirements*. For example, section 3.3., *Usability Requirements*, requires descriptions of the usability, quality requirements, and objectives while using the software. In addition, those requirements can include measurable criteria from perspectives such as effectiveness.

The standard structure can be seen as defining the essential components that should be included in an SRS. In other words, it can serve as a reference for evaluating the quality of an SRS. Thitisathienkul and Prompoon have proposed a method for assessing the quality of SRSs based on the presence or absence of topics corresponding to each section in the IEEE 830's standard structure [10]. Their approach quantifies the quality by linking the occurrence of sections to quality attributes to be satisfied. Similarly, Takoshima and Aoyama have defined an inspection matrix that associates sections of the standard structure with quality attributes and assessed practical SRSs using their framework [11]. To accommodate structural differences among SRSs, they also introduced a translation matrix, which maps the sections of a given SRS to those of the standard structure. However, this matrix must be manually created for each SRS, making it costly to implement.

LLMs like GPT-4o[2] are language models trained with a vast number of texts. They have demonstrated high capabilities in NLP tasks such as text generation and translation. Research integrating LLMs into requirements engineering is on the rise [4], and some studies relate to specifications. Krishna et al. have evaluated LLMs' capability in creating and modifying SRSs and investigated the effort reduction achieved through LLM use [12]. They reported that although the generated SRSs lacked detailed descriptions, they achieved a quality comparable to those of entry-level software engineers. Based on these results, they concluded that utilizing LLMs could enhance productivity. Norheim and Rebentisch have attempted to structure requirements written in natural language using LLMs [13]. Their approach transforms requirements into requirement template structures like EARS [14] and semi-formal structures like linear temporal logic. They reported that while traditional rule-based methods required numerous custom rules, LLMs could perform the transformation with only a few examples. From these studies, we can glimpse the potential of applying LLMs to tasks such as generating requirement specifications and transforming requirement descriptions. Since the structural standardization of SRS can be viewed as a task of regenerating its content in a different structural format, it is also reasonable to explore the applicability of LLMs to this task.

[1]https://doi.org/10.5281/zenodo.15743755

[2]https://openai.com/index/hello-gpt-4o/

**1 Introduction**
- 1.1 Purpose
- 1.2 Scope
- 1.3 Product overview
  - 1.3.1 Product perspective
  - 1.3.2 Product functions
  - 1.3.3 User characteristics
  - 1.3.4 Limitations
- 1.4 Definitions

**2 References**

**3 Requirements**
- 3.1 Functions
- 3.2 Performance requirements
- 3.3 Usability requirements
- 3.4 Interface requirements
- 3.5 Logical database requirements
- 3.6 Design constraints
- 3.7 Software system attributes
- 3.8 Supporting information

**4 Verification**
- (parallel to subsections in Section 3)

**5 Appendices**
- 5.1 Assumptions and dependencies
- 5.2 Acronyms and abbreviations

Fig. 1. The section structure presented in ISO/IEC/IEEE 29148 [5]

## III. Structural standardization of SRS

This study aims to transform an SRS with any given structure into the standard structure. A standardized SRS should satisfy two requirements: it retains all the necessary and sufficient information from the original, and its information is appropriately arranged according to the standard. In other words, the transformation should standardize only the structure while preserving the original content.

As the first step toward achieving this goal, we leverage an LLM to standardize SRSs and assess their effectiveness. Based on the two properties, we define the following two research questions (RQs) and aim to answer them:

**RQ1** *To what extent does the SRS restructured using an LLM retain information from the original SRS?*

**RQ2** *To what extent does the SRS restructured using an LLM adhere to the standard structure and descriptions?*

## IV. Experimental design

Figure 2 shows the flow of our experiment. We give an LLM a target SRS and ISO 29148 to obtain a standardized SRS. We repeat it five times to address the probabilistic instability of the

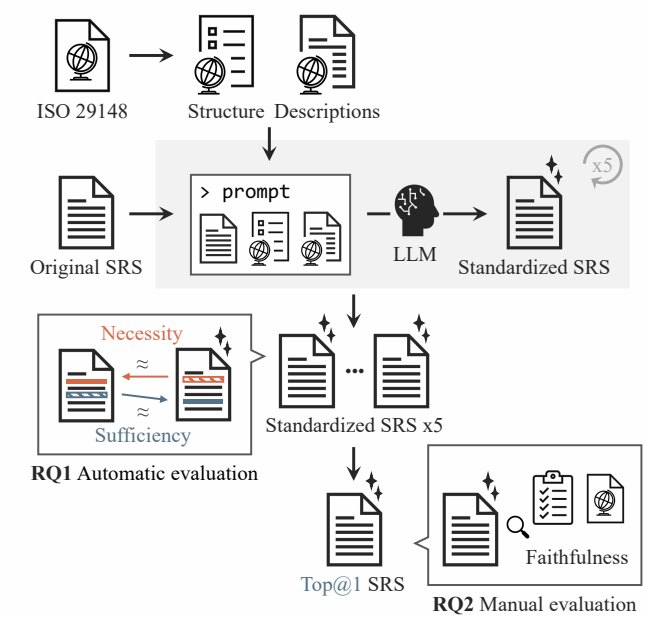


Fig. 2. The overview of our experiment

LLM's outputs. For each standardized SRS, we automatically evaluate the *necessity* and *sufficiency* of the standardized using metrics defined later. Afterward, we select the highest sufficiency and check its faithfulness to the standard with our checklist. We use `gpt-4o-2024-11-20`[3], which is the latest model in the cost-effective and high-performance GPT-4o models.

### A. Prompts design

What information we should give to the LLM is an original SRS, the overall of standard structure, and the descriptions of required content for each section. We should provide the overall structure to ensure that each section is appropriately balanced and organized. For example, although section 1.3.2, *Product functions*, and section 3.1, *Function*, in ISO 29148 describe the same concept, *functions*, each has different granularity. Section 1.3 is *Product overview*, while section 3 is *Requirements*. In other words, section 1.3.2 is about more abstract features of functions than section 3.1.

We have designed two types of prompts in this study. One is the most straightforward prompt, which gives all the information and lets the LLM standardize an SRS simultaneously. We refer to this prompt as $\texttt{P}_{\texttt{all}}$. The other is a prompt that lets the LLM standardize an SRS for each section. We give an SRS, the overall standard structure, which section to be generated, and descriptions of the section. This simple segmentation reduces the number of tokens required for generation and can improve the accuracy of the outputs. After all sections are generated, we merge them into a standardized SRS. We refer to this prompt strategy as $\texttt{P}_{\texttt{each}}$. Both prompts contain the instruction, "Retain necessary and sufficient information from the original SRS". This is because one of the purposes of this experiment is to evaluate generated SRSs from the viewpoint of information retention. Both also contain "Output `[N/A]` for any required content missing from the original SRS". This instruction aims to determine whether the LLM can recognize that certain items required in standardized SRSs are absent in their originals. Additionally, as a postprocessing step for subsequent evaluation, we insert any required sections the LLM failed to generate, with their content set to "`[NULL]`". We distinguish items the LLM overlooked from those that were initially absent and indicated as "`[N/A]`".

### B. Dataset

We use some SRSs in PURE dataset [15] for this experiment. Table I shows the SRSs with the number of words and sentences after a preprocessing explained later. These statistics were generated using the `word_tokenize` and `sent_tokenize` from NLTK[4]. There are three criteria for selecting the subjects. The first is that it declares itself as SRS. Since PURE includes specifications that declare themselves as FRS (Functional Requirements Specification) or SyRS (System Requirements Specification), we exclude them. The second is that it does not contain any figures. In this experiment, we target SRSs that consist of text only. (10) contains a figure, but it is only displayed on the title page without a caption. The figure does not provide any additional information to readers. Therefore, we include (10). The third is that it does not contain tables other than revision histories. Unlike sentences, tables are read without a fixed direction. The automatic evaluation process involves segmenting texts. Tables are unsuitable for this evaluation because they lack unique segmentation. However, revision histories are essential content and cannot be excluded. They are typically represented as tables with dates and version numbers as primary keys. Thus, they can be converted into bullet lists.

As mentioned earlier, we preprocessed all the target SRSs. First, we excluded the cover page, table of contents, footers, and headers. This information is unnecessary for the LLM to regenerate SRSs. Next, we punctuated all bullet items to segment them into sentences for the automatic evaluation. If an item was a complete sentence, we placed a period at the end of the line; otherwise, we placed a comma. The last item was followed by a period regardless of its content. This punctuation was applied recursively in the case of nested.

[3]https://platform.openai.com/docs/models/gpt-4o

[4]https://www.nltk.org/

TABLE I
SRS SUBJECTS IN PURE DATASET [15]

| Name (abbr.) | | #words | #sents |
|---|---|---|---|
| 2001 - libra | (01) | 5,235 | 206 |
| 2003 - qheadache | (03) | 2,321 | 170 |
| 2007 - get real 0.2 | (07g) | 2,564 | 112 |
| 2007 - puget sound | (07p) | 4,064 | 187 |
| 2008 - vub | (08) | 8,353 | 417 |
| 2009 - library | (09) | 4,389 | 158 |
| 2010 - home 1.3 | (10) | 4,528 | 158 |

### C. Evaluation for RQ1

We define the metrics *necessity* and *sufficiency* to quantitatively assess the extent to which standardized SRSs retain the semantic information of the original. Let $S_{\text{std}}$ and $S_{\text{orig}}$ be the sets of sentences from a standardized and an original SRSs, respectively. The necessity, $\text{nec}(S_{\text{std}}, S_{\text{orig}})$, and sufficiency, $\text{suff}(S_{\text{std}}, S_{\text{orig}})$, are formulated as follows:

$$\text{nec}(S_{\text{std}}, S_{\text{orig}}) = \underset{s' \in S_{\text{orig}}}{\text{mean}} \underset{s \in S_{\text{std}}}{\max} \text{similarity}(s, s'),$$

$$\text{suff}(S_{\text{std}}, S_{\text{orig}}) = \text{nec}(S_{\text{orig}}, S_{\text{std}}).$$

Intuitively, $\text{nec}(S_{\text{std}}, S_{\text{orig}})$ captures the degree to which the content of $S_{\text{orig}}$ is semantically preserved in $S_{\text{std}}$. A higher value indicates that more information from $S_{\text{orig}}$ is present in $S_{\text{std}}$. Conversely, $\text{suff}(S_{\text{std}}, S_{\text{orig}})$ measures the extent to which the content of $S_{\text{std}}$ is supported by $S_{\text{orig}}$. These metrics can be regarded as a sentence-level extension of BERTScore [16]. By considering $S_{\text{orig}}$ as the reference and $S_{\text{std}}$ as the candidate, $\text{nec}(S_{\text{std}}, S_{\text{orig}})$ and $\text{suff}(S_{\text{std}}, S_{\text{orig}})$ correspond to recall and precision, respectively.

We implement $\text{similarity}(s, s')$ as the cosine similarity between sentence embeddings generated by SBERT [17]. To obtain these embeddings, we use the pre-trained `all-MiniLM-L12-v2`[5] model. As a preprocessing step, we remove lines containing `[N/A]` or `[NULL]`, and sections consisting solely of such lines. This is because items judged to be "not present" are often not explicitly stated in original SRSs. For example, a document without references will typically not include a statement such as "There are no references". Consequently, even when the LLM correctly identifies an absent item by outputting `[N/A]`, a direct comparison may lead to an unfairly low evaluation score. After this preprocessing, we segment SRSs into sentences using `sent_tokenize`.

There is no direct numerical interpretation of the cosine similarity with SBERT. Therefore, we construct a simple mapping between values and interpretations using STSB (Semantic Textual Similarity Benchmark) dataset[6]. STSB dataset was initially created by Cer et al. It consists of pairs of English sentences labeled on a six-point scale from 0 to 5 based on their semantic relatedness [18]. For example, a sentence pair labeled as 4 is judged as "The two sentences are mostly equivalent, but some unimportant details differ", 3 as "The two sentences are roughly equivalent, but some important information differs/missing", and 2 as " The two sentences are not equivalent, but share some details". We compute the cosine similarity for each pair in the dataset and fit a linear regression model to predict the normalized label. The left plot in Figure 3 shows the scatter plot of the cosine similarity and normalized label for each pair of sentences. The right shows the residual plot. As a result of fitting, the regression coefficient was about 0.9655, and the intercept was about $-0.08175$. The coefficient of determination, i.e., R-squared, was approximately 0.721. We interpret the necessity and sufficiency values based on the estimated labels. In detail, the necessity label is calculated as $y_{\text{nec}} = 5 \cdot (-0.08175 + 0.9655 \cdot \text{nec}(S_{\text{std}}, S_{\text{orig}}))$. The sufficiency label, $y_{\text{suff}}$, is calculated in a similar manner.

[5]https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2

[6]https://huggingface.co/datasets/sentence-transformers/stsb

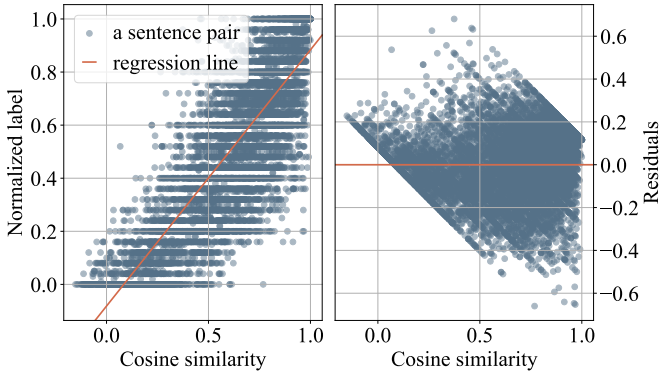


Fig. 3. Linear regression using STSB dataset[6]. The left figure shows the scatter plot of the cosine similarity and normalized label, including the regression line. The right figure shows the residual plot.

### D. Evaluation for RQ2

We check whether generated SRSs contain items that should be described. We make a checklist about which items should be written in standardized SRSs, referring to a formulation by Boyarchuk et al. [19]. They formulated items required for each standard section as sets. For example, about S_12 corresponding to section 3.3, *Usability*, S_12 = {ubr, mec, mefc, msc}. Each element is an item that is simplified descriptions from ISO 29148, such as "ubr: usability requirements" and "mec: measurable criteria of effectiveness in specific use contexts".

However, we made two modifications when creating the checklist. First, we added several items. The formulation by Boyarchuk et al. does not include sections 1.4, 2, and 5.2 of the standard structure. Therefore, based on ISO 29148, we formulated new items as 1.4 = {dbd}, 2 = {lrd, tndo, srof}, and 5.2 = {aa}, where "dbd: definitions for any words or phrases that have meaning beyond dictionaries", "lrd: list of referred documents", "tndo: document title, report number, date and publishing organization", "srof: sources which the references can be obtained from", and "aa: acronyms and abbreviations used in the documents". Second, we organize the items hierarchically. Some items have an inclusion relation with each other. For example, about ubr and mec in S_12, ubr can include mec. In other words, mec may be described as a specific instance of ubr. Figure 4 shows the trees representing hierarchical relations among some items. Since both eso and ras can be included simultaneously in faapi and fapgo, we splitted eso and ras and place under faapi and fapgo accordingly. The suffixes "_i" and "_o" indicate items related to input and output. Items not included in Figure 4 are independent.

Among the five standardized SRSs, we select one with the highest sufficiency for check. This check assumes that information in standardized SRSs is included in their original. Even when standardized SRSs include descriptions of specific items, they may not be present in or may differ from their

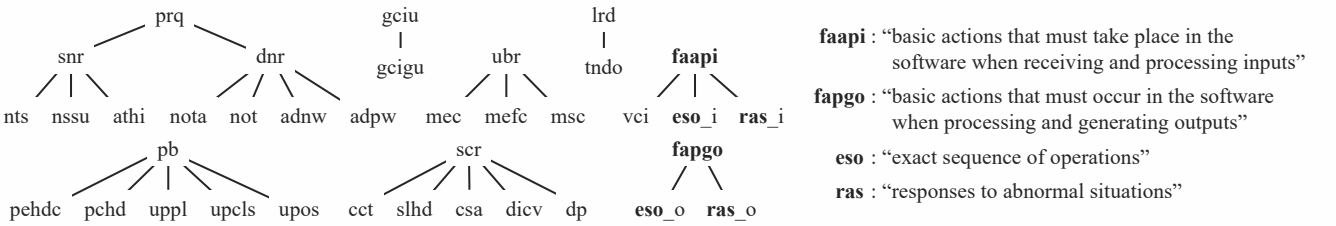


Fig. 4. Trees representing hierarchical relations among items formulated by Boyarchuk et al. [19] and us. All items not shown are independent.

original. To reduce the risk of such cases, i.e., hallucination, we select the SRS with the highest sufficiency for the check. However, we note that hallucinations cannot be completely ruled out in this way.

This check will be conducted independently by two authors. The final score is the proportion of TRUE in the checklist. We determine the truth value of an item by the logical OR of its check and the truth values of its sub-items. For example, if a check is made for mec, mec is TRUE. Then, its parent item, ubr, is also considered TRUE, regardless of whether it is checked. Items marked with `[N/A]` are checked and treated as TRUE, whereas items marked with `[NULL]` are not checked.

## V. Experiment

### A. An example of standardization

Figure 5 presents an example of a standardization application. The target SRS to be standardized was (01). From the standardized SRSs, section 3.3, *Usability requirements*, are extracted and shown. The original SRS did not contain a section explicitly labeled as "Usability requirements". Therefore, fragments considered to be related to standard section 3.3 are extracted and shown from the original SRS.

The standardized with $\texttt{P}_{\texttt{all}}$ is concise. Its content does not contradict the original SRS. However, it lacks details and does not contain descriptions that could be interpreted as usability requirements, such as response time. Additionally, the required content for standardized SRSs described in ISO 29148 is not fully covered. On the other hand, the standardized with $\texttt{P}_{\texttt{each}}$ is described in more detail than that with $\texttt{P}_{\texttt{all}}$. Furthermore, it includes bullet points for each required item described in ISO 29148, making it more faithful to the standard. However, it also contains descriptions not explicitly stated in the original. For example, statements like "The interface should provide clear feedback on job status and scheduling decisions" go beyond mere functional descriptions and were not found in the original. Based on these characteristics, $\texttt{P}_{\texttt{all}}$ and $\texttt{P}_{\texttt{each}}$ are expected to exhibit opposite properties in terms of the evaluation criteria of this study.

### B. Results for RQ1

Table II shows the results for each original SRS and prompt. $y_{\text{nec}}$ and $y_{\text{suff}}$ are the necessity and sufficiency label for a standardized SRS. $\mathbf{y}_{\text{nec}}/y_{\text{suff}}$ column represents the pair of $y_{\text{nec}}$ and $y_{\text{suff}}$ when $y_{\text{nec}}$ was the highest among the five standardized. Conversely, $y_{\text{nec}}/\mathbf{y}_{\text{suff}}$ column represents the pair when $y_{\text{suff}}$ was the highest. $\bar{y}_{\text{nec}}$ and $\bar{y}_{\text{suff}}$ are the means of $y_{\text{nec}}$ and $y_{\text{suff}}$, respectively.

As an overall trend, $\texttt{P}_{\texttt{each}}$ shows the higher necessity but the lower sufficiency compared to $\texttt{P}_{\texttt{all}}$. This trend indicates that standardized SRSs generated by $\texttt{P}_{\texttt{each}}$ contained more information from the originals than those generated by $\texttt{P}_{\texttt{all}}$, whereas the content produced by $\texttt{P}_{\texttt{all}}$ was more frequently included in the originals. It may be partly attributed to the size of standardized SRSs. The standardized by $\texttt{P}_{\texttt{all}}$ consisted of about $1,000$ words on average, whereas those by $\texttt{P}_{\texttt{each}}$ consisted of about $5,000$ words, making it roughly five times longer. Since $\texttt{P}_{\texttt{all}}$ produced shorter standardized SRSs than their originals, it inevitably functioned as summaries, resulting in lower necessity but higher sufficiency. Conversely, $\texttt{P}_{\texttt{each}}$ exhibited the opposite trend. The excessive length allowed the standardized SRSs to include information from the originals, but it also introduced redundant descriptions, resulting in a higher necessity but lower sufficiency. These observations suggest that the size of a standardized SRS relative to its original impacts the necessity and sufficiency. It also highlights the importance of prompts in appropriately balancing the size of the generated SRSs.

From an absolute perspective, the results cannot be considered entirely satisfactory. Across all SRSs and prompts, the necessity label had a maximum average of approximately 3.35, and the sufficiency label had about 3.25. These scores suggest that while one set of information was mainly present

TABLE II
THE NECESSITY AND SUFFICIENCY LABEL FOR EACH SRS AND PROMPT. $\mathbf{y}_{\text{nec}}/y_{\text{suff}}$ COLUMN SHOWS THE PAIR OF $y_{\text{nec}}$ AND $y_{\text{suff}}$ WHEN $y_{\text{nec}}$ WAS AT ITS MAXIMUM. $y_{\text{nec}}/\mathbf{y}_{\text{suff}}$ DOES THE SAME FOR $y_{\text{suff}}$. $\bar{y}_{\text{nec}}$ AND $\bar{y}_{\text{suff}}$ SHOW THEIR MEAN VALUES.

| SRS | Prompt | $\mathbf{y}_{\text{nec}}/y_{\text{suff}}$ | $y_{\text{nec}}/\mathbf{y}_{\text{suff}}$ | $\bar{y}_{\text{nec}}$ | $\bar{y}_{\text{suff}}$ |
|---|---|---|---|---|---|
| (01) | $\texttt{P}_{\texttt{all}}$ | **2.70** / 3.36 | 2.70 / **3.36** | 2.57 | 3.25 |
| | $\texttt{P}_{\texttt{each}}$ | **3.37** / 3.10 | 3.35 / **3.18** | 3.35 | 3.13 |
| (03) | $\texttt{P}_{\texttt{all}}$ | **2.28** / 3.01 | 2.24 / **3.11** | 2.17 | 3.04 |
| | $\texttt{P}_{\texttt{each}}$ | **3.07** / 3.07 | 3.07 / **3.07** | 3.05 | 3.02 |
| (07g) | $\texttt{P}_{\texttt{all}}$ | **2.43** / 2.83 | 2.43 / **2.87** | 2.38 | 2.78 |
| | $\texttt{P}_{\texttt{each}}$ | **2.63** / 2.46 | 2.60 / **2.46** | 2.62 | 2.46 |
| (07p) | $\texttt{P}_{\texttt{all}}$ | **2.47** / 3.32 | 2.47 / **3.32** | 2.42 | 3.25 |
| | $\texttt{P}_{\texttt{each}}$ | **3.07** / 2.82 | 3.03 / **2.84** | 3.00 | 2.77 |
| (08) | $\texttt{P}_{\texttt{all}}$ | **2.03** / 3.14 | 1.99 / **3.26** | 2.00 | 3.18 |
| | $\texttt{P}_{\texttt{each}}$ | **2.67** / 3.05 | 2.65 / **3.05** | 2.66 | 3.02 |
| (09) | $\texttt{P}_{\texttt{all}}$ | **1.83** / 2.71 | 1.78 / **2.74** | 1.79 | 2.70 |
| | $\texttt{P}_{\texttt{each}}$ | **2.74** / 2.92 | 2.73 / **2.94** | 2.66 | 2.85 |
| (10) | $\texttt{P}_{\texttt{all}}$ | **2.41** / 3.10 | 2.41 / **3.10** | 2.37 | 3.02 |
| | $\texttt{P}_{\texttt{each}}$ | **3.12** / 2.91 | 3.12 / **2.91** | 3.08 | 2.83 |

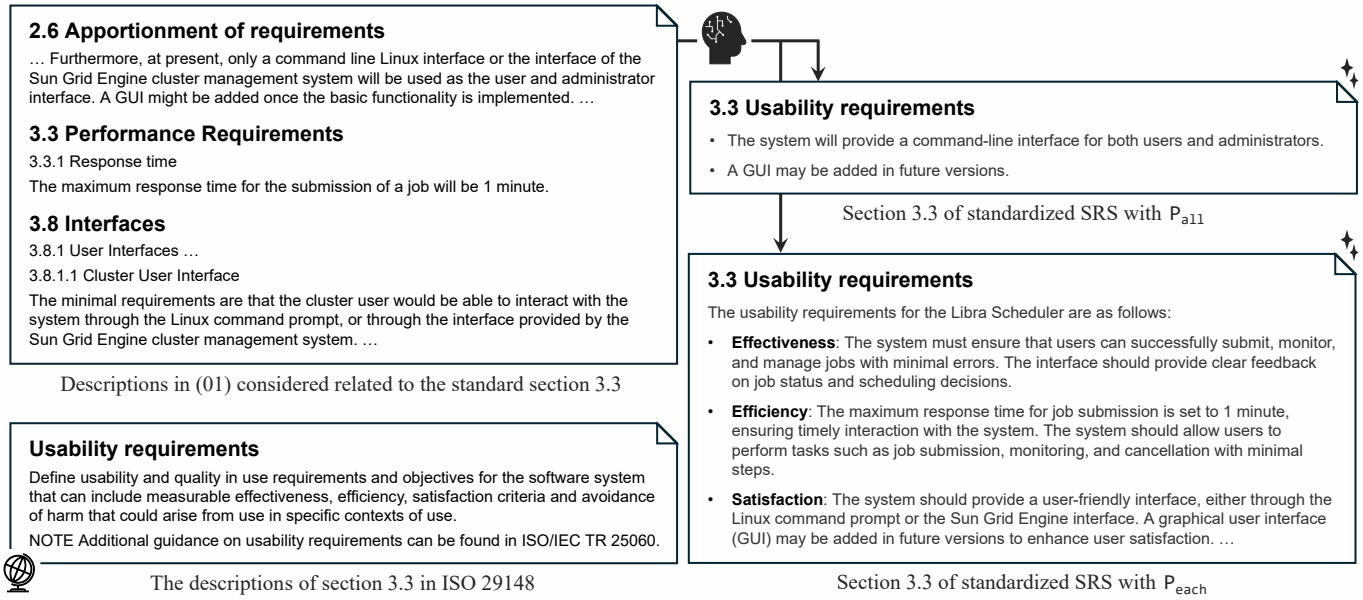


Fig. 5. An example of a standardization application.

in the other, some important content was missing. In particular, the necessity label for $\texttt{P}_{\texttt{all}}$ was low; for (09), the maximum was only 1.83. It indicates that the standardized SRSs did not semantically include much of the original content except for shared topics and details. These findings suggest that while LLM-standardized SRSs retain some information in their originals, important parts may be lost. Moreover, the relatively low sufficiency scores imply that information not present in the original SRSs may have been introduced. Standardized SRSs generated by the LLM can not be ready for use without further review or correction.

**An answer to RQ1**: In $\texttt{P}_{\texttt{all}}$, information in standardized SRSs tends to be contained within their original SRSs, whereas in $\texttt{P}_{\texttt{each}}$, the opposite is true. However, in both prompts, it cannot be said that standardized SRSs sufficiently and necessarily retain information from their original SRSs.

### C. Results for RQ2

Table III shows the check rates for the standardized SRSs with the highest sufficiency in each SRS and prompt. The checks were performed by two authors. The table presents each result along with their average and Cohen's $\kappa$.

$\texttt{P}_{\texttt{each}}$ had almost double the check rate of $\texttt{P}_{\texttt{all}}$. In $\texttt{P}_{\texttt{all}}$, the check rate was approximately 40%, whereas in $\texttt{P}_{\texttt{each}}$, it exceeded 80%. Therefore, $\texttt{P}_{\texttt{each}}$ was more successful in faithfully standardizing SRSs according to the descriptions of the standard and the standard structure.

However, we found several errors even in the standardized SRSs generated by $\texttt{P}_{\texttt{each}}$. For example, section 1.1, *Purpose*, should describe the purpose of the software according to the descriptions of the standard, section 1.1. However, in several cases, the purpose of the SRS was described instead of that of the software. When checking the original SRS, we found a section titled "Purpose", which contained the SRS's purpose. This suggests that the section's naming in original SRSs led to describing content that differed from what should have been written. Additionally, we found inconsistencies between section 3.1, *Function*, and the corresponding section 4. The information items in section 4 are recommended to be given for corresponding to those in section 3. Even though, in section 3.1, item "ras: responses to abnormal situations" was marked as `[N/A]`, there were descriptions of verification about ras in section 4. These issues highlight the need for prompts that correctly guide the recognition of what should be described and consider the dependencies between the sections.

**An answer to RQ2**: Standardized SRSs generated by $\texttt{P}_{\texttt{each}}$ covered about 80% of the items that should be included in the standard structure. However, while the items were covered, the content often differed from the intended description, and inconsistencies were observed between sections.

## VI. Discussion

### A. Evaluation for RQ1

We defined and measured the *necessity* and *sufficiency* to quantify how standardized SRSs retain the semantic information of their originals well. We also built a regression model to map these values to human interpretations, thereby addressing RQ1. However, there are three major issues with this approach. In this section, we discuss these issues and potential solutions.

The first issue concerns the sentence-level evaluation. For example, when calculating the *necessity*, we select the best-matching sentence in a standardized SRS for each sentence in

the original. However, the meaning of a sentence is not always fully conveyed in isolation; its surrounding context is often necessary for accurate interpretation. Our metric can be viewed as an extension of BERTScore [16] from the token level to the sentence level. In BERTScore, tokens are not treated independently, as they are encoded with awareness of other tokens in the sentence via the attention mechanism [20]. As a result, BERTScore incorporates intra-sentence context even though it performs matching at the token embedding level. In contrast, SBERT encodes each sentence independently [17], and therefore, our implementation does not account for inter-sentence context. To address this issue, sentence decontextualization, as proposed by Choi et al. [21], offers a promising solution. This technique enables individual sentences to be interpreted independently while preserving their original meaning and context. By applying it, sentence-based computations can effectively incorporate contextual information.

The second issue lies in the similarity-based evaluation. These metrics rely on the existence of semantically equivalent sentences, which can lead to an underestimation of the degree of information preserved from one to the other. From the perspective of the *necessity* and *sufficiency*, we can accept not only semantic similarity but also asymmetric entailment relations. For example, in the calculation of the *sufficiency*, it should be acceptable if a sentence in a standardized SRS is an abstraction of a sentence in the original. Any omitted content in such cases can be identified through the calculation of the *necessity*, which evaluates the reverse direction. Capturing asymmetric entailment relations between sentences aligns with tasks such as Natural Language Inference (NLI) and Fact Verification. Large-scale datasets such as SNLI [22], MNLI [23], and FEVER [24] have been released, and numerous high-performing methods have been proposed [25][26]. The usefulness of NLI in requirements analysis has also been demonstrated [27]. By leveraging these techniques, we can more accurately assess the extent to which information is retained in each direction.

The third issue concerns the difficulty of interpreting the metrics. We applied linear regression using STSB dataset to interpret the metric values. However, the effectiveness of this approach is limited. The right panel of Figure 3 shows the residual plot. Since the labels are originally discrete values on a six-point scale, the residuals exhibit a downward trend. The coefficient of determination is $0.72$, indicating that the regression does not fully capture the underlying relationship. In addition, because the labels are on an ordinal scale, interpreting the regression outputs remains challenging. To enhance interpretability, the metric should be derived from surface-level syntactic information rather than latent semantic features. For example, a statement such as "80% of the sentences in the standardized SRS can be inferred from the original SRS" would provide a more intuitive and easily interpretable measure of information retention.

Addressing these issues is crucial for accurately evaluating the results of structural standardization. Furthermore, our evaluation strategy may also be helpful for quality assessment in other tasks, such as generating SRS documents from unstructured requirements like user stories using LLMs. By implementing and refining the metrics and clarifying their practical impact, this work is expected to contribute to establishing a more reliable and objective evaluation framework.

TABLE III
THE CHECK RATES AND COHEN'S $\kappa$ FOR EACH SRS AND PROMPT

| SRS | Prompt | Author A | Author B | Mean | $\kappa$ |
|---|---|---|---|---|---|
| (01) | $\mathtt{P_{all}}$ | 43.0 % | 53.8 % | 48.4 % | .660 |
| | $\mathtt{P_{each}}$ | 78.5 % | 84.9 % | 81.7 % | .500 |
| (03) | $\mathtt{P_{all}}$ | 37.6 % | 49.5 % | 43.5 % | .677 |
| | $\mathtt{P_{each}}$ | 65.6 % | 74.2 % | 69.9 % | .696 |
| (07g) | $\mathtt{P_{all}}$ | 35.5 % | 44.1 % | 39.8 % | .688 |
| | $\mathtt{P_{each}}$ | 79.6 % | 86.0 % | 82.8 % | .775 |
| (07p) | $\mathtt{P_{all}}$ | 46.2 % | 50.5 % | 48.4 % | .656 |
| | $\mathtt{P_{each}}$ | 75.3 % | 84.9 % | 80.1 % | .701 |
| (08) | $\mathtt{P_{all}}$ | 59.1 % | 64.5 % | 61.8 % | .750 |
| | $\mathtt{P_{each}}$ | 84.9 % | 89.2 % | 87.1 % | .524 |
| (09) | $\mathtt{P_{all}}$ | 35.5 % | 48.4 % | 41.9 % | .609 |
| | $\mathtt{P_{each}}$ | 73.1 % | 86.0 % | 79.6 % | .420 |
| (10) | $\mathtt{P_{all}}$ | 40.9 % | 41.9 % | 41.4 % | .756 |
| | $\mathtt{P_{each}}$ | 82.8 % | 86.0 % | 84.4 % | .715 |

### B. Evaluation for RQ2

We evaluated the faithfulness of standardized SRSs to the standard using the checklist to verify the presence or absence of the required items. However, this evaluation did not consider the relationship between standardized SRSs and their originals, which may lead to an overestimation in RQ2. LLMs may enhance conformity to the standard, potentially resulting in undesirable outputs. One such issue is hallucination, where content not present in the original SRS is generated. Conversely, LLMs may excessively abstract the original content, leading to a failure to fully reproduce the intended requirements. Therefore, to properly assess the structural standardization capability of LLMs, the evaluation must consider not only the standardized SRSs but also their original counterparts.

Additionally, we need to check for inconsistencies between sections within a standardized SRS. Some of the standardized SRSs exhibited contradictions across different sections. However, this issue was not considered in the evaluation of RQ2, which may have led to an overestimation of the results. Inconsistencies among requirements are a well-known challenge in requirements engineering, and many studies have addressed this issue [27][28][29]. Such inconsistencies must also be taken into account when evaluating the structural standardization of SRSs. Incorporating inter-sectional consistency checks into future evaluation frameworks is essential for a more accurate assessment of the quality of standardized SRSs.

The evaluations in this experiment were conducted solely by the two authors, raising concerns about potential subjective bias. While Cohen's $\kappa$ exceeded 0.6 for 11 items, it fell below 0.6 for three items and was as low as 0.42 for (09) $\mathtt{P_{each}}$. According to the criteria proposed by Landis and Koch [30], although most items demonstrated substantial agreement, some cannot be regarded as having been evaluated consistently. Moreover, since the evaluators were the authors themselves,

there is a risk that unconscious bias may have influenced the results. To ensure the reliability of the evaluation, it is essential to incorporate assessments by multiple independent third-party evaluators.

### C. Generalizability

In this study, we utilized GPT-4o and SRSs from PURE dataset. Different experimental results may be obtained when using other LLMs or SRSs. In particular, the prompts employed in this experiment are tailored to the current experimental setup. LLMs have limitations on token length. For instance, GPT-4o has a context window of 128,000 tokens and a maximum output length of 16,384 tokens[3]. The SRSs used in this study are relatively short, up to approximately 8,500 words, and thus easily fit within these limits. However, real-world SRSs can be significantly longer, and in such cases, the prompts used in this study, $\texttt{P}_{\texttt{all}}$ and $\texttt{P}_{\texttt{each}}$, may not be effective.

### D. Restructuring of SRS

The structural standardization of SRSs presents several potential issues in itself. One such issue is that not all information in original SRSs can necessarily be fully mapped onto the standardized structure. This study assumes that each statement in an original SRS can be mapped to one of the sections in a standardized SRS. However, given the diversity of SRS formats, this assumption does not always hold. Standardization may inevitably lead to the loss of certain information, potentially resulting in incomplete requirement definitions. Moreover, transforming original SRSs may have adverse effects on related artifacts. A typical example is traceability links. If links have already been established based on original SRSs, structural standardization may render them invalid. The loss of traceability can hinder change management and impact analysis, thereby compromising maintainability.

These issues can cause significant problems in software development and, therefore, require mitigation. Although manual evaluation is currently the only practical option, there are ways to reduce the associated effort. For example, the evaluation strategy based on the *necessity* and *sufficiency*, as used in RQ1, can be leveraged. This strategy constructs mappings between two SRSs by measuring the semantic similarity between individual sentences. By utilizing this information, it becomes possible to identify information lost during standardization, i.e., descriptions unique to an original SRS. Furthermore, since the mapping reveals which sentences in a standardized SRS correspond to those in the original, it can also support the referencing and restoration of traceability links that were initially established in the original. Other techniques, such as NLI, can also be effective for aligning information between pre- and post-standardization SRSs.

## VII. Future works

We have three main future challenges: expanding the analysis of the experimental results, engineering prompts, and extending modalities. From an analytical perspective, it is necessary to focus on the influence of the original SRSs' structure and their domains. In this study, we used seven different SRSs as experimental subjects, but the impact of their individual characteristics has not been considered. Analyzing this aspect is significant because it supports one of the key advantages of SRS standardization: the ability to uniformly utilize SRSs with different structures.

It is also necessary to design prompts that incorporate both prompt techniques and characteristics of this task. In this study, we designed straightforward prompts, $\texttt{P}_{\texttt{all}}$ and $\texttt{P}_{\texttt{each}}$. However, various prompt techniques have been proposed to improve the generation accuracy of LLMs, such as few-shot learning [31][32], Chain of Thought [33], and Retrieval-Augmented Generation [34]. In systems utilizing LLMs, applying these techniques is now a prerequisite and essential for achieving practical performance. In addition, approaches specific to SRS standardization should also be considered. An approach is to ensure an appropriate length for standardized SRSs. SRS standardization is not a general summarization task and should not abstract or omit information from original SRSs. As one of its necessary conditions, standardized SRSs are expected to have a physical length equal to or greater than their originals. Another approach is to account for dependencies between sections. Some standardized SRSs generated using $\texttt{P}_{\texttt{each}}$ were inconsistencies between section 3.1, *Functions* and section 4, *Verification*. Such inconsistencies within a standardized SRS can be addressed by designing recursive prompts based on section dependencies.

In this study, we focused on SRSs composed solely of texts. However, in actual SRSs, it is common to use figures and tables to aid readers in understanding. It is valuable to design prompts and methods that can handle SRSs containing figures and tables and evaluate their effectiveness to assess the applicability of LLM-based standardization to a broader range of SRS types. Recent LLM models like GPT-4o can accept not only text but also images[2]. Therefore, it would be important to investigate the applicability of the simple prompts designed in this study to SRSs that include images.

## VIII. Conclusion

In this study, we attempted to transform the section structure of any SRS into the section structure recommended by the international standard ISO/IEC/IEEE 29148. As an approach to restructuring, we utilized an LLM with strong NLP capabilities and evaluated the standardization ability using simple prompts. Although the method in this study does not ensure that information in original SRSs is fully preserved, standardized SRSs cover approximately 80% of the items that should be present in the standard structure and can be considered to have been faithfully transformed.

***Acknowledgment***

This work was partly supported by JSPS KAKENHI Grant Number 25K15056.

## REFERENCES


[1] C. Denger and T. Olsson, *Quality Assurance in Requirements Engineering*. Springer, 2005, pp. 163–185.

[2] T. Tamai and M. I. Kamata, "Impact of Requirements Quality on Project Success or Failure," in *Design Requirements Engineering: A Ten-Year Perspective*. Springer, 2009, pp. 258–275.

[3] L. Zhao, W. Alhoshan, A. Ferrari, K. Letsholo, M. Ajagbe, E.-V. Chioasca, and R. Batista-Navarro, "Natural Language Processing for Requirements Engineering: A Systematic Mapping Study," *Trans. Computing Surveys*, vol. 54, no. 3, pp. 1–41, 2021.

[4] H. Cheng, J. Husen, Y. Lu, T. Racharak, N. Yoshioka, N. Ubayashi, and H. Washizaki, "Generative AI for Requirements Engineering: A Systematic Literature Review," 2025, arXiv preprint. [Online]. Available: https://arxiv.org/abs/2409.06741

[5] "ISO/IEC/IEEE International Standard - Systems and Software Engineering – Life Cycle Processes – Requirements Engineering," *ISO/IEC/IEEE 29148:2018(E)*, pp. 1–104, 2018.

[6] "IEEE Recommended Practice for Software Requirements Specifications," *IEEE Std 830-1998*, pp. 1–40, 1998.

[7] X. Franch, M. Glinz, D. Mendez, and N. Seyff, "A Study About the Knowledge and Use of Requirements Engineering Standards in Industry," *Trans. Software Engineering*, vol. 48, no. 9, pp. 3310–3325, 2022.

[8] M. Aoyama and T. Nakane, "ReqQA: A Software Requirements Specifications Quality Analyzer and Its Application," *Trans. Information Processing Society of Japan*, vol. 57, no. 2, pp. 694–706, 2016, (in Japanese).

[9] A. Chikh and M. Aldayel, "A New Traceable Software Requirements Specification Based on IEEE 830," in *Proc. International Conference on Computer Systems and Industrial Informatics (ICCSII)*, 2012, pp. 1–6.

[10] P. Thitisathienkul and N. Prompoon, "Quality Assessment Method for Software Requirements Specifications Based on Document Characteristics and Its Structure," in *Proc. International Conference on Trustworthy Systems and Their Applications (TSA)*, 2015, pp. 51–60.

[11] A. Takoshima and M. Aoyama, "A Two-Stage Inspection Method for Automotive Software Systems and Its Practical Applications," in *Proc. International Requirements Engineering Conference (RE)*, 2016, pp. 313–322.

[12] M. Krishna, B. Gaur, A. Verma, and P. Jalote, "Using LLMs in Software Requirements Specifications: An Empirical Evaluation," in *Proc. International Requirements Engineering Conference (RE)*, 2024, pp. 475–483.

[13] J. Norheim and E. Rebentisch, "Structuring Natural Language Requirements with Large Language Models," in *Proc. International Requirements Engineering Conference Workshops (REW)*, 2024, pp. 68–71.

[14] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak, "Easy Approach to Requirements Syntax (EARS)," in *Proc. International Requirements Engineering Conference (RE)*, 2009, pp. 317–322.

[15] A. Ferrari, G. Spagnolo, and S. Gnesi, "PURE: A Dataset of Public Requirements Documents," in *Proc. International Requirements Engineering Conference (RE)*, 2017, pp. 502–505.

[16] T. Zhang, V. Kishore, F. Wu, K. Weinberger, and Y. Artzi, "BERTScore: Evaluating Text Generation with BERT," in *Proc. International Conference on Learning Representations (ICLR)*, 2020.

[17] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019, pp. 3982–3992.

[18] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation," in *Proc. International Workshop on Semantic Evaluation (SemEval)*, 2017, pp. 1–14.

[19] A. Boyarchuk, O. Pavlova, M. Bodnar, and I. Lopatto, "Approach to the Analysis of Software Requirements Specification on Its Structure Correctness," in *Proc. International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS)*, 2020, pp. 85–95.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. ukasz Kaiser, and I. Polosukhin, "Attention is All you Need," in *Proc. Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[21] E. Choi, J. Palomaki, M. Lamm, T. Kwiatkowski, D. Das, and M. Collins, "Decontextualization: Making Sentences Stand-Alone," *Trans. Association for Computational Linguistics*, vol. 9, pp. 447–461, 2021.

[22] S. Bowman, G. Angeli, C. Potts, and C. Manning, "A Large Annotated Corpus for Learning Natural Language Inference," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015, pp. 632–642.

[23] A. Williams, N. Nangia, and S. Bowman, "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference," in *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2018, pp. 1112–1122.

[24] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "FEVER: A Large-scale Dataset for Fact Extraction and VERification," in *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2018, pp. 809–819.

[25] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," 2019, arXiv preprint. [Online]. Available: https://arxiv.org/abs/1907.11692

[26] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," in *Proc. International Conference on Neural Information Processing Systems (NeurIPS)*, 2019, pp. 5753–5763.

[27] M. Fazelnia, V. Koscinski, S. Herzog, and M. Mirakhorli, "Lessons from the Use of Natural Language Inference (NLI) in Requirements Engineering Tasks," in *Proc. International Requirements Engineering Conference (RE)*, 2024, pp. 103–115.

[28] V. Bertram, H. Kausch, E. Kusmenko, H. Nqiri, B. Rumpe, and C. Venhoff, "Leveraging Natural Language Processing for a Consistency Checking Toolchain of Automotive Requirements," in *Proc. International Requirements Engineering Conference (RE)*, 2023, pp. 212–222.

[29] M. Fazelnia, M. Mirakhorli, and H. Bagheri, "Translation Titans, Reasoning Challenges: Satisfiability-Aided Language Models for Detecting Conflicting Requirements," in *Proc. International Conference on Automated Software Engineering (ASE)*, 2024, pp. 2294—-2298.

[30] R. Landis and G. Koch, "The Measurement of Observer Agreement for Categorical Data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977.

[31] Brown, T. et al., "Language Models Are Few-Shot Learners," in *Proc. Conference on Neural Information Processing Systems (NeurIPS)*, 2020, pp. 1877–1901.

[32] T. Kojima, S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large Language Models Are Zero-Shot Reasoners," in *Proc. Conference on Neural Information Processing Systems (NeurIPS)*, 2022, pp. 22 199–22 213.

[33] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in *Proc. Conference on Neural Information Processing Systems (NeurIPS)*, 2022, pp. 24 824–24 837.

[34] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," 2024, arXiv preprint. [Online]. Available: https://arxiv.org/abs/2312.10997